ORIGINAL

Brent O. Hatch (5715)
Mark F. James (5295)
HATCH, JAMES & DODGE
10 West Broadway, Suite 400
Salt Lake City, Utah 84101
Telephone: (801) 363-6363
Facsimile: (801) 363-6666

Robert Silver (admitted pro hac vice)
Edward Normand (admitted pro hac vice)
Sean Eskovitz (admitted pro hac vice)
BOIES, SCHILLER & FLEXNER LLP
333 Main Street
Armonk, New York 10504
Telephone: (914) 749-8200
Facsimile: (914) 749-8300

Stuart H. Singer (admitted pro hac vice)
BOIES, SCHILLER & FLEXNER LLP
401 East Las Olas Boulevard – Suite 1200
Ft. Lauderdale, Florida 33301
Telephone: (954) 356-0011
Facsimile: (954) 356-0022

Stephen N. Zack (admitted pro hac vice)
BOIES, SCHILLER & FLEXNER LLP
Bank of America Tower – Suite 2800
100 Southeast Second Street
Miami, Florida 33131
Telephone: (305) 539-8400
Facsimile: (305) 539-1307

Attorneys for The SCO Group, Inc.

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF UTAH

THE SCO GROUP, INC. Plaintiff/Counterclaim-Defendant, v. INTERNATIONAL BUSINESS MACHINES CORPORATION, Defendant/Counterclaim-Plaintiff.	**DECLARATION IN SUPPORT OF SCO'S OPPOSITION TO IBM'S CROSS-MOTION FOR PARTIAL SUMMARY JUDGMENT** **[Docket No. 197]** **(REFILED IN REDACTED FORM)** Case No. 2:03CV0294DAK Honorable Dale A. Kimball Magistrate Judge Brooke C. Wells

DECLARATION OF SANDEEP GUPTA

1. My name is Sandeep Gupta and I am employed by The SCO Group, Inc. My office is located at 430 Mountain Avenue, Murray Hill, NJ 07974. Unless otherwise noted or evident from the context, this declaration is based on my personal knowledge and information available to me from reliable sources. To the best of my knowledge, information and belief, the facts set forth herein are true and correct.

2. I submit this Declaration in support of the SCO's Opposition to IBM's Cross-Motion for Partial Summary Judgment, in the lawsuit entitled The SCO Group, Inc. v. IBM, Civil No. 2:03-CV-0294 DAK (D. Utah 2003).

3. In this declaration, I will explain why I believe that several routines and several groupings of code for which SCO has copyright protection were copied into the Linux operating system. Specifically, this declaration will (1) describe how the Read-Copy-Update routine in Linux is substantially similar to a routine in UNIX; (2) describe how the user level synchronization (ULS) routines in Linux are substantially similar to routines in UNIX; (3) describe how Linux version 2.4.20 contains code that is either an identical or substantially similar copy of SCO's UNIX System V IPC code; (4) identify identical and substantially similar copying of SCO's copyrighted UNIX "header and interfaces" in Linux; (5) describe how Linux version 2.6 contains code that is an identical copy of SCO's UNIX System V init (SYS V init) code; and (6) identify identical copying of SCO's UNIX Executable and Linking Format (ELF) code in Linux.

The declaration will discuss these topics of copyright infringement in the order just described. Although I am not an expert in copyright law, I believe these topics either show copying of code or raise significant factual issues that need to be explored further.

4. SCO claims ownership of copyrights to UNIX software, source code, object code, programming tools, documentation related to UNIX operating system technology, and derivative works thereof. These materials are covered by numerous copyright registrations issued by the United States Copyright Office (the "Copyright Registrations"). These registrations have been obtained by SCO and its predecessors in interest and are owned by SCO. Included among such registrations are:

	TITLE	REG. NO.	REG. DATE
1	UNIX Operating System Edition 5 and Instruction Manual	TXu-510-028	03/25/92
2	UNIX Operating System Edition 6 and Instruction Manual	TXu-511-236	04/07/92
3	UNIX Operating System Edition 32V and Instruction Manual	TXu-516-704	05/15/92
4	UNIX Operating System Edition 7 and Instruction Manual	TXu-516-705	05/15/92
5	Operating System Utility Programs	TXu-301-868	11/25/87
6	UNIXWARE 7.1.3	TX-5-787-679	06/11/03
7	UNIX SYSTEM V RELEASE 3.0	TX-5-750-270	07/07/03
8	UNIX SYSTEM V RELEASE 3.1	TX-5-750-269	07/07/03
9	UNIX SYSTEM V RELEASE 3.2	TX-5-750-271	07/07/03
10	UNIX SYSTEM V RELEASE 4.0	TX-5-776-217	07/16/03
11	UNIX SYSTEM V RELEASE 4.1IES	TX-5-705-356	06/30/03
12	UNIX SYSTEM V RELEASE 4.2	TX-5-762-235	07/03/03
13	UNIX SYSTEM V RELEASE 4.1	TX-5-762-234	07/03/03
14	UNIX SYSTEM V RELEASE 3.2	TX-5-750-268	07/09/03
15	UNIX SYSTEM V RELEASE 4.2 MP	TX 5-972-097	6/29/2004

5. I believe that the Read-Copy-Update ("RCU") routine in Linux version 2.6.5 and in patches to Linux (hereinafter referred to as Linux RCU) are substantially similar to the RCU version in SCO's copyrighted work, specifically UNIX SVR4.2 MP. UNIX SVR4.2 MP is part of a registered copyright of SCO entitled UNIX System V Release 4.2 MP (also known as 4.2 ES/MP) (Registration No. TX 5-972-097).

6. A problem may occur in a multiprocessing computer environment when multiple entities such as multiple processors attempt to access data in a shared memory. For example, a user of the data, such as a "process" in the operating system, may attempt to update a particular piece of data at the same time that another process is attempting to read the same data. In this scenario, the process attempting to read the data will see the data in a partially updated state, or in a non-updated state, rather than in the updated state being supplied by the other process. To address this situation, programmers have devised various synchronization methods that allow access to data by multiple processors at the same time, and that maintain synchronization by directing all processes -- at the appropriate time -- to the updated data.

7. RCU (Read-Copy-Update) is one of the methods used to synchronize access to shared data in a multiprocessing environment.

8. Because RCU provides for synchronized access to shared data in a multiprocessing environment, RCU provides substantial performance

enhancements to an operating system and is thus a very important part of any multiprocessing operating system.

9. SCO's version of RCU first appeared in UNIX SVR4.2 MP and is referred to herein as UNIX RCU.

10. The Linux operating system includes a routine, Linux RCU, that is substantially similar to UNIX RCU.

11.

REDACTED

12. There are several ways to synchronize data updating and sharing without using the five acts just described. In other words, synchronizing access to shared data

in a multiprocessing environment can be expressed in ways other than the expressions in UNIX RCU.

13. For example, synchronizing access to shared data can be expressed with mutual exclusion locks (mutex). Mutual exclusion locks may be used to lock a data location so that only one operating system process can have read/write access to that data location at one time. If another operating system process needs to access the same data location, the second operating process has to wait until the data location has been unlocked. Using mutual exclusion locks, access to shared data is seriatim rather than simultaneous. Synchronizing access to shared data can also be expressed with reader/writer locks. Using reader/writer locks, multiple processes can acquire read access to a data location, but only one process at a time can acquire write access to update that data location.

14. UNIX RCU and Linux RCU perform the same five acts in the same sequence, and UNIX RCU and Linux RCU have the same structure and organization.

15. Each of the five acts of the UNIX RCU -- and of the Linux RCU -- routine is expressed in one or a few lines of code.

16. The first act, "allocating a new data structure of a certain size," is expressed in UNIX RCU and Linux RCU by a single line of nearly identical code. From a software programmer's perspective, the UNIX RCU expression of the act of allocating a new data structure has been identically copied into Linux RCU. As can be seen in attached Exhibit A, the Linux RCU code (column 4) for the first act is nearly identical to the UNIX RCU code (column 1) for the first act.

17.

REDACTED

18.

REDACTED

19.

REDACTED

20.

REDACTED

21. In Linux RCU, in contrast, the fifth act of deferred deletion is achieved by a callback function that is automatically called when no current users remain so that the old data structure may be deleted.

22. Although the fifth act in Linux RCU and in UNIX RCU are expressed differently, they both achieve deferred deletion of the old data structure.

23. In my opinion, Linux RCU is substantially similar to UNIX RCU, and appears to be derived from UNIX RCU.

24. As represented to me, contributors to Linux had access to UNIX RCU. Showing access to UNIX RCU is a two step process. First, UNIX RCU was copied into Dynix, which is Sequent's version of UNIX, and then UNIX RCU

was released from Dynix into Linux. Sequent Computer System, Inc.'s (Sequent) software engineers could have accomplished the first step -- of developing a Dynix RCU based on the UNIX RCU -- when they worked under the Multiprocessor Software Cooperation Agreement (the "MP Agreement" – Exhibit B) with Unix System Laboratories, Inc. (USL) engineers to develop the UNIX version of RCU. The MP Agreement was signed in September of 1990. USL is a predecessor of SCO.

25. Jack Slingwine and Paul McKenney are the credited authors of Dynix RCU, and were both Sequent employees. At least Mr. Slingwine was involved in the UNIX development work under the MP Agreement. At least Mr. Slingwine had access to the UNIX RCU work because of his involvement in the UNIX development work. I believe that Mr. Slingwine would have used that access to UNIX development to review UNIX RCU because of his clear interest in RCU. Regarding his clear interest in RCU, Mr. Slingwine and Mr. McKenney authored a paper on RCU, "Read-Copy Update : Using Execution History To Solve Concurrency Problems," which refers to "work performed at Sequent." *See* Exhibit C. In this paper, Mr. Slingwine and Mr. McKenney thank (among others) Brent Kingsbury, and Mr. Kingsbury was one of the authors of a design document for UNIX which discusses, among other things, UNIX RCU.

26. As represented to me, evidence that the Sequent RCU work was performed during the same time frame that the MP Agreement was operative is clear. Specifically in that regard, the MP Agreement, signed in September 1990,

appears to terminate in November 1992. It took two years to develop UNIX RCU under the MP Agreement, from late 1990 to late 1992. On July 19, 1993, a patent application related to RCU entitled "Apparatus and Method for Achieving Reduced Overhead Mutual Exclusion and Maintaining Coherency in a Multiprocessor..." was filed listing Mr. Slingwine and Mr. McKenney as co-inventors. The patent issued on August 19, 1995 as U.S. Patent No. 5,442,758 and lists Sequent as the assignee. *See* Exhibit D.

27. In sum, at least Mr. Slingwine (and perhaps Mr. McKenney) had access to UNIX developments during the USL/Sequent collaboration under the MP Agreement, which included development of UNIX RCU, and both showed great interest in RCU by filing a patent application (as co-inventors) relating to RCU immediately after the MP Agreement collaboration.

28. Thus, I believe that UNIX RCU was copied into another version of UNIX known as Dynix by engineers who worked for Sequent at the time and who later worked for IBM when IBM acquired Sequent.

29. As far as the second step, IBM thereafter released Dynix, with a copied and modified UNIX RCU in Dynix, into Linux. More specifically, the Dynix version of RCU was used by IBM employee (and former Sequent employee) Dipankar Sarma to create a software patch for placing a substantially similar version of RCU into Linux. I believe that the first patch appears to be for Linux version 2.4.1 and was contributed by Mr. Sarma. *See* Exhibit E. A paper entitled "Read-Copy Update" also lists Mr. Sarma along with Mr. McKenney

and others as authors. *See* Exhibit F. Another patch was also provided to Linux version 2.5.44 by IBM employee Mingming Cao. *See* Exhibit G. This patch appears to be incorporated into the Linux version 2.6.

30. I will now describe the user level synchronization (ULS) routines that are used for blocking and unblocking of processes.

REDACTED

31. The ULS routines in Linux are commonly referred to as FUTEX (Fast User Mutex), but will be called Linux ULS here.

32. The ULS routines in UNIX are sometimes referred to as usync, but will be called UNIX ULS here.

33. The main purpose of the *ULS* routines is to facilitate inter-process synchronization by blocking and unblocking processes attempting to access shared data.

34. Synchronization of user level processes accessing shared data is important to prevent two processes from modifying the same data at the same time. The blocking and unblocking of access to shared data by ULS allows only one process at a time to use the shared data. ULS is a very significant piece of code because it is a very important part of any operating system to synchronize access by processes to shared data.

35. SCO's version of ULS first appeared in UNIX SVR4.2 MP.

36. **REDACTED**

REDACTED

37. REDACTED

38.

REDACTED

39. **REDACTED**

40.

REDACTED

41.

REDACTED

42.

REDACTED

43.

REDACTED

44.

REDACTED

45.

REDACTED

46.

REDACTED

47. There are several ways to implement the blocking and unblocking for ULS. Therefore, the result of ULS can be achieved in several other ways.

REDACTED

48. Another alternative implementation of blocking and unblocking for ULS is the Linux implementation that existed prior to Linux copying the UNIX ULS implementation. This alternative ULS implementation is described in the article attached as Exhibit K. In this article, Rusty Russell, of IBM, describes how the earlier Linux ULS implementation was improved by Jamie Lokier and Hugh Dickins in the new Linux ULS implementation.

49. While working on changing the Linux ULS, Mr. Lokier learned of the UNIX ULS implementation, may have had access to it, and may have incorporated the

infringing code into Linux. Some evidence of this is the changelog to the current Linux ULS code which designates Mr. Lokier as author of the Linux ULS code. Mr. Lokier's author's notes in the changelog give special thanks to Mr. Russell of IBM and to Hugh Dickins (a former SCO employee) for help with the patch to the Linux ULS code which Mr. Lokier authored. I believe that the collaboration between Jamie Lokier and Hugh Dickins to improve the Linux ULS code benefited from access to and knowledge of the UNIX ULS implementation.

50. I now turn to instances of copying in Linux of UNIX System V IPC routines.

REDACTED

51. IPC stands for Inter-Process Communication.

REDACTED

52.

REDACTED

53. With regard to the organization of Linux SysVIPC and UNIX System V IPC, both consist of three mechanisms: message queues, semaphore, and shared memory. There is no reason for the organization to be identical other than the fact that Linux SysVIPC has been copied from UNIX System V IPC.

REDACTED

54.

REDACTED

55.

REDACTED

56.

TABLE 1

REDACTED

57.

TABLE 2

REDACTED

58.

TABLE 3

REDACTED

59.

TABLE 4

REDACTED

60.

TABLE 5

REDACTED

61.

TABLE 6

REDACTED

62. SCO owns valid copyrights for UNIX System V IPC header file code. As can be seen from Tables 1-6, SCO's UNIX System V IPC header file code has been identically copied into Linux. Slight variations in some of the terms are insubstantial differences in programming. As to access, UNIX System V IPC header file code was included in any system running a UNIX System V derived operating system. Therefore, anyone having access to a system installed with UNIX System V or one or more of its derivatives could have had access to UNIX System V IPC header file code.

63. I will now describe certain UNIX System V headers and interfaces that are copied either identically or substantially similarly in Linux.

64. Regarding access to this header and interface code, UNIX header and interface source code is available in SCO-copyrighted documentation, such as manual pages. These manual pages are published on the Internet with copyright notices. Also, UNIX header and interface code is available to any entity having a license to UNIX, or who can otherwise access the UNIX code.

65. A header file is a programming source file containing declarations of interfaces that facilitate communication between different regular source files that comprise the program. The interfaces can come from the program itself, or from libraries. The libraries provide (define) the interfaces that are intended to be used elsewhere, typically in an application.

66. **REDACTED**

67.

REDACTED

68.

REDACTED

69. REDACTED

70.

REDACTED

71.

REDACTED

72

REDACTED

73. I now turn to instances of copying in Linux of SCO's copyrighted SYS V init code. Linux version 2.6 contains code that is an identical copy of SYS V init code.

74. SYS V init was accessible for copying because the manual pages defining SYS V init features, for example init and inittab, are published as electronic documents and are available to anyone with an Internet browser. These manual pages, however, carry appropriate copyright notices. Using the manual pages, a skilled programmer could copy the structure, sequence, and organization of SYS V init routines. SYS V init and inittab were included in documentation with each release of UNIX System V as manual pages init(1M) and inittab(4), respectively. *See* Exhibits Z and AA. Also, SYS V init code is available to any entity having a license to UNIX, or who can otherwise access the UNIX code.

75.

REDACTED

76.

REDACTED

77. I will now describe certain UNIX Executable and Linking Format (ELF) code that is copied either identically or substantially similarly in Linux.

78. The ELF code is used for processing executables and object code.

REDACTED

In the code presented in Tables 7 - 11, comments (i.e., text beginning with "/*" and ending with "*/") and other irrelevant information have been omitted for clarity. Also, in some cases, the code lines and/or code sections have been reordered for convenience. The code with comments and other information can be seen in the attached Exhibits CC (UNIX code) and DD (Linux code).

79.

REDACTED

80.

TABLE 7

REDACTED

REDACTED

81. The data shown in Tables 8 -11 are additional examples of copying of UNIX code in Linux.

82.

TABLE 8

REDACTED

83.

TABLE 9

REDACTED

84.

TABLE 10

REDACTED

85.

TABLE 11

REDACTED

86.

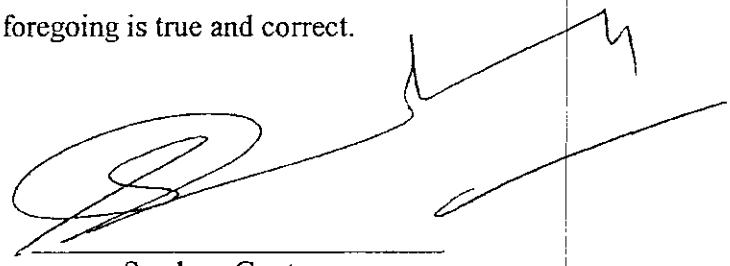
REDACTED

Moreover, the portions of Linux that are direct copies of UNIX are the meaningful, functional portions of the code

REDACTED

I declare under penalty of perjury that the foregoing is true and correct.

July 7, 2004



Sandeep Gupta

CERTIFICATE OF SERVICE

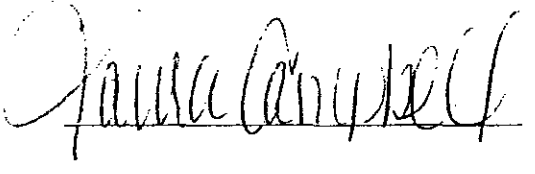
Plaintiff, The SCO Group, hereby certifies that a true and correct copy of **DECLARATION IN SUPPORT OF SCO'S OPPOSITION TO IBM'S CROSS-MOTION FOR PARTIAL SUMMARY JUDGMENT** was served on Defendant International Business Machines Corporation on the 9th day of July, 2004, as follows:

BY HAND DELIVERY:

Alan L. Sullivan, Esq.
Todd M. Shaughnessy, Esq.
Snell & Wilmer L.L.P.
15 West South Temple, Ste. 1200
Salt Lake City, Utah 84101-1004

Evan R. Chesler, Esq.
Cravath, Swaine & Moore LLP
825 Eighth Avenue
New York, NY 10019

Donald J. Rosenberg, Esq.
1133 Westchester Avenue
White Plains, New York 10604



CERTIFICATE OF SERVICE

Plaintiff/Counterclaim Defendant, The SCO Group, Inc., hereby certifies that a true and correct copy of the foregoing was served on Defendant IBM on the 5th day of July, 2005 by U.S. Mail to:

David Marriott, Esq.
CRAVATH SWAINE & MOORE LLP
Worldwide Plaza
825 Eighth Avenue
New York, NY 10019

Donald Rosenberg, Esq.
1133 Westchester Avenue
White Plains, NY 10604

Todd Shaughnessy, Esq.
SNELL & WILMER LLP
1200 Gateway Tower West
15 West South Temple
Salt Lake City, UT 84101-1004

Laura K. Chaves